

Apprentissage automatique

Séance 1 --- Introduction et les données

Halim Djerroud



révision : 23/09/2026

Organisation du cours

Les 6 séances :

- ➊ **Introduction et les données**
(importer, examiner)
- ➋ Préparer les données
(visualiser, nettoyer, encoder, normaliser)
- ➌ Régression linéaire simple et multiple
- ➍ Classification et clustering
- ➎ Réseaux de neurones
(perceptron, MLP, rétropropagation)
- ➏ Réseaux de neurones avec Keras

Déroulement :

- 18 heures : 6 séances de 3 heures
- Chaque séance : 1 h 30 de cours, 1 h 30 de TP
- Deux contrôles continus (QCM)
- Un projet à faire en binôme
- Un examen écrit

Prérequis

Python de base. Aucune connaissance préalable en statistiques n'est supposée.

Au programme aujourd'hui

Cours (1 h 30)

- ❶ Qu'est-ce que l'apprentissage automatique ?
- ❷ Les données : nature, types, sources, qualité
- ❸ Les 9 étapes de la préparation des données
- ❹ Importer des données
- ❺ Examiner la structure d'un dataset

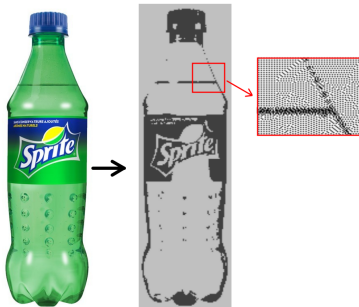
TP (1 h 30)

- Installation de l'environnement (Python, Jupyter, Pandas, Scikit-learn)
- Chargement du dataset *California Housing*
- Examen de sa structure
- Comparaison avec le dataset original

Question de départ

Pourquoi ne peut-on pas simplement *écrire* un programme qui reconnaît un spam ?

Exemple 1 : écrire un programme qui reconnaît une bouteille



```
if(img[x,y] == PLEIN) && ...  
if(img[x,y+1] == PLEIN) && ...  
if(img[x,y+2] == VIDE) && ...  
...  
{  
    return TRUE;  
}  
else{  
    return FALSE;  
}
```

À vous

À votre avis, est-il possible d'écrire ce programme ? Combien de `if` faudrait-il pour couvrir toutes les bouteilles, sous tous les angles, sous tous les éclairages ?

Exemple 2 : écrire un programme qui reconnaît les spams

To: Undisclosed recipients: ;
From: UNIVERSITY WEBMAIL SYSTERM <kjjuranko@verizon.net>
Date: 07/12/2008 06:42PM
Subject: Attn University Webmail System User

Dear Customer,

Our investigation of spam complaints shows that your email address is compromised and was used to send spam message through our University Webmail System. Your Username will be **disable** if you did not fill the form below and send it back to us.

CUSTOMER BILLING FORM

First Name.:
Last Name.:
Student ID.:
Date Of Birth.:
Email Address.:
PASSWORD.:
RETYPE PASSWORD.:

Thank you,
University Information Technology Security Office

Annotations:
- "SYSTERM" and "kjjuranko@verizon.net" are marked as "Unknown sender".
- "disable" is marked as "Watch for grammatical errors".
- The billing form fields are marked as "will never ask for this information".
- "University Information Technology Security Office" is marked as "No contact info".

```
if(is_authorized_senders(list)) && .  
if(text_content(list_words) &&...  
if(spell-check(email)<0.8) && ...  
...  
{  
    return SPAM;  
}  
else{  
    return NOT-SPAM;  
}
```

Le vrai problème

Supposons que vous y arriviez. Demain, les spammers écrivent V1AGRA au lieu de

Pourquoi ces programmes sont impossibles à écrire

❶ Trop de cas à énumérer

Une bouteille peut être vue sous une infinité d'angles, de lumières, de fonds. Aucune liste de règles n'est exhaustive.

❷ On ne sait pas expliciter la règle

Vous reconnaissez une bouteille en un dixième de seconde, mais vous êtes incapable d'écrire la règle que vous venez d'appliquer.

❸ Le problème change avec le temps

Les spammeurs s'adaptent. Un programme figé devient faux tout seul.

Le renversement

Puisqu'on ne sait pas **écrire** les règles, on va les faire **découvrir** à la machine, à partir d'**exemples**.

Programmation classique

règles + données → résultats

Apprentissage automatique

données + résultats → **règles**

Définir l'apprentissage : T, E, P

Tom Mitchell, *Machine Learning*, 1997

Un programme **apprend** de l'expérience **E**, pour une tâche **T**, mesurée par une performance **P**, si sa performance sur **T**, mesurée par **P**, **s'améliore** avec **E**.

Filtrer les spams

- **T** : classer un mail en spam / non-spam
- **E** : des milliers de mails déjà étiquetés
- **P** : le pourcentage de mails bien classés

Estimer le prix d'un logement

- **T** : prédire un prix
- **E** : des logements vendus, avec leur prix
- **P** : l'écart moyen entre prix prédit et prix réel

Pourquoi cette définition

Elle est **opérationnelle**. Devant n'importe quel problème, posez-vous les trois questions : quel est mon **T** ? mon **E** ? mon **P** ? Si vous ne savez pas y répondre, vous n'avez pas encore un problème d'apprentissage.

À quoi ressemble un problème d'apprentissage ?

X --- les caractéristiques				y --- la cible
MedInc	HouseAge	AveRooms	...	MedHouseVal
8.3252	41.0	6.98	...	4.526
8.3014	21.0	6.24	...	3.585
7.2574	52.0	8.29	...	3.521

- Une **ligne** = un **exemple** (une observation)
- Une **colonne** = une **caractéristique**
- n = nombre d'exemples, p = nombre de caractéristiques

Apprendre, c'est trouver une fonction

$$f : X \longrightarrow y$$

qui se trompe le moins possible sur des exemples **nouveaux**.

Les grandes familles d'apprentissage

Supervisé — on dispose de y

- **Régression** : y est une **valeur continue**
prix d'un logement, température
- **Classification** : y est une **catégorie**
spam / non-spam, espèce d'iris, tumeur
bénigne / maligne

Non supervisé — pas de y

- **Partitionnement** : regrouper ce qui se
ressemble
segmenter des clients
- **Réduction de dimension** : résumer
l'information

Le reste du cours sur cette carte

- **Séance 3** — régression
- **Séance 4** — classification et
partitionnement
- **Séances 5 et 6** — réseaux de neurones,
qui font les deux

La question à se poser en premier

**Est-ce que je dispose de la réponse pour
mes exemples ?**

Oui → supervisé. Non → non supervisé.
Tout le reste en découle.

Comment les programmes apprennent ? --- le principe

- L'apprentissage se fait à partir de la **reconnaissance de motifs** dans les données.
- On utilise des **modèles** qui ajustent leurs **paramètres** pour mieux prédire ou classifier de nouvelles données.
- L'objectif n'est pas de bien répondre sur les données déjà vues, mais de **généraliser** à des données nouvelles.

L'analogie de l'examen

Réviser sur les annales, c'est l'**entraînement**. Faire un sujet blanc pour s'auto-évaluer, c'est la **validation**. Le jour J, le sujet que vous découvrez, c'est le **test**.

Un étudiant qui apprend les annales par cœur sans les comprendre a une note parfaite à l'entraînement, et échoue le jour J : c'est exactement ce qu'on cherche à éviter avec un modèle.

Entraînement, validation, test

	Validation	Test
Entraînement $\approx 70\%$	15%	15%

- Ajuster les **paramètres** du modèle
- Choisir entre plusieurs modèles, régler les **hyperparamètres**
- Mesurer la performance **finale**, une seule fois

La règle d'or

Le jeu de **test** ne sert **qu'une seule fois**, tout à la fin. Si vous l'utilisez pour choisir votre modèle, il devient un jeu de validation, et vous n'avez plus aucune estimation honnête de vos performances réelles.

Comment les programmes apprennent ? --- le schéma

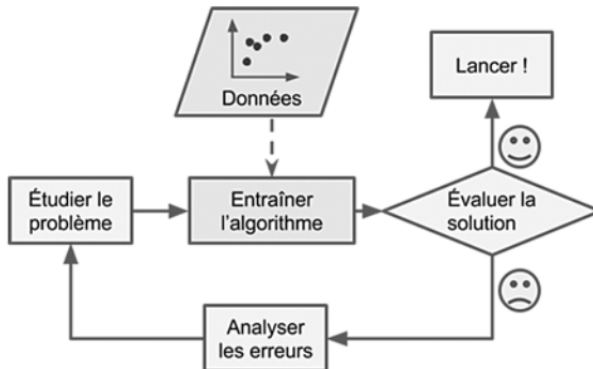


Figure: Source : livre *Machine Learning avec Scikit-Learn*

Où se situe l'apprentissage automatique dans l'IA ?

IA Statistique (Machine Learning) :

- Apprentissage à partir de données
- Reconnaissance de motifs
- Approches probabilistes
- **Exemples :**
 - Réseaux de neurones, MLP
 - Arbres de décision
 - Machines à vecteurs de support

IA Symbolique :

- Raisonnement logique
- Représentation explicite des connaissances
- Règles et symboles
- **Exemples :**
 - Systèmes experts
 - Systèmes multi-agents
 - Planification automatique

Dans ce cours

Nous nous concentrerons sur l'**IA statistique**. Remarquez que l'IA symbolique, elle, repose précisément sur les règles écrites à la main dont on vient de montrer les limites --- les deux approches sont complémentaires.

L'apprentissage automatique autour de vous

Vous l'utilisez déjà aujourd'hui :

- Filtrage des spams de votre boîte mail
- Suggestions de Netflix, Spotify, YouTube
- Reconnaissance vocale de votre téléphone
- Traduction automatique
- Détection de fraude bancaire
- Aide au diagnostic médical

Ce que vous saurez faire dans 6 séances

- Prédire le prix d'un logement à partir de ses caractéristiques
- Classer une fleur, une tumeur, un passager du Titanic
- Regrouper des clients par comportement
- Construire et entraîner un réseau de neurones

Et surtout : savoir **si** votre modèle marche vraiment.

Les données

Les Données

Objectifs du chapitre :

- 1 Comprendre l'importance des données dans les projets d'IA.
- 2 Identifier les différentes sources et types de données.
- 3 Apprendre les étapes du traitement des données avant l'entraînement des modèles.
- 4 Maîtriser les bonnes pratiques en matière de gestion des données.

Qu'est-ce qu'une donnée ?

- **Définition** : Informations **brutes**, **quantitatives** ou **qualitatives**, utilisées pour entraîner un modèle d'IA.
- **Caractéristiques des bonnes données** :
 - **Pertinence** : Doit être liée à la tâche à accomplir.
 - **Fiabilité** : Doit être exempté de bruit ou d'erreurs.
 - **Représentativité** : Couvrir l'ensemble des cas possibles.
 - **Quantité** : Un volume suffisant de données est nécessaire.

Données Brutes, Quantitatives et Qualitatives

1. Données Brutes : Données collectées directement, non traitées, contenant souvent du bruit, des doublons ou des valeurs manquantes.

- **Exemples** : Relevés de capteurs, fichiers CSV bruts, vidéos de surveillance.
- **Problèmes** : Données manquantes, anomalies, bruit.
- **Solution** : Nettoyage, normalisation, filtrage.

2. Données Quantitatives : Données numériques permettant des calculs mathématiques (discrètes ou continues).

- **Exemples** : Température (36,5°C), nombre de clics, distances (2,35 m).
- **Sous-types** :
 - **Discrètes** : Nombre d'enfants (1, 2, 3, ...).
 - **Continues** : Température, distance, durée (2,35 min).

Données Brutes, Quantitatives et Qualitatives

3. Données Qualitatives : Données non numériques qui classent ou catégorisent des objets.

- **Exemples** : Couleurs (rouge, bleu), avis clients (positif, neutre, négatif), type d'appareil (smartphone, tablette).
- **Types** :
 - **Catégoriques** : Aucune notion d'ordre (ex : couleur, genre).
 - **Ordinales** : Classées par ordre (ex : avis client : mauvais, moyen, bon).

Comparatif

- **Données Brutes** : Non traitées, bruitées, non nettoyées.
- **Données Quantitatives** : Numériques, mesurables (discrètes ou continues).
- **Données Qualitatives** : Catégories, classes ou labels (ordinales ou nominales).

Types et Sources de Données

Types de Données

- **Données Numériques** : Température, distance, age.
- **Données Catégoriques** : Genre, statut (actif/inactif), labels de classes.
- **Données Textuelles** : Avis clients, résumés de textes.
- **Données Visuelles** : Images, vidéos.
- **Séries Temporelles** : Prix des actions, données de capteurs.

Sources de Données

- **Sources Publiques** : Kaggle, UCI Machine Learning Repository.
- **Sources Internes** : Bases de données d'entreprise, CRM, ERP.
- **Données Simulées** : Données générées par des simulations.
- **Données du Web** : Collecte (web scraping) respectant les règles RGPD.

Qualité des Données

● Problèmes courants :

- Données manquantes.
- Valeurs aberrantes.
- Données dupliquées.
- Biais des données.

● Solutions :

- Imputation des valeurs manquantes.
- Filtrage des valeurs aberrantes.
- Suppression des doublons.

Les 9 étapes de la préparation des données

1. **Importer les données** : fichiers CSV, bases de données, API.
2. **Examiner la structure** : types des colonnes, valeurs manquantes, statistiques.
3. **Explorer et visualiser** : histogrammes, boîtes à moustaches, nuages de points.
4. **Nettoyer** : valeurs manquantes, valeurs aberrantes, doublons.
5. **Gérer les variables qualitatives** : One-Hot Encoding, Label Encoding.
6. **Normaliser et mettre à l'échelle** : standardisation, normalisation.
7. **Rechercher des corrélations** : matrice de corrélation.
8. **Combiner des variables** : feature engineering, réduction de dimension (ACP).
9. **Séparer entraînement / validation / test**.

Étapes 1 et 2 : aujourd'hui. Étapes 3 à 9 : séance 2.

Éthique et RGPD

Pourquoi l'éthique des données est essentielle ?

Garantir une IA fiable, équitable et conforme aux lois en protégeant les utilisateurs.

- **Protection des données personnelles**

- Collecte minimale (principe de minimisation).
- Consentement explicite et utilisation transparente.

- **Anonymisation et Sécurisation**

- Pseudonymisation, floutage, agrégation.
- Stockage sécurisé (chiffrement, contrôle d'accès).

- **Équité et absence de biais**

- Détection et réduction des biais dans les données.
- Évaluation régulière des impacts sur les utilisateurs.

- **Traçabilité et responsabilité**

- Documenter l'origine des données et les transformations.
- Garantir l'auditabilité des modèles et des décisions.

Importer les données

Importer les données

Objectifs du chapitre :

- 1 Les différentes façons d'importer des données
- 2 Utiliser des dataset existants

Pandas : Introduction et Import de Données

Bibliothèque d'analyse de données

Pandas permet de manipuler et analyser des données structurées (tableaux, séries temporelles)

Import depuis différentes sources

```
import pandas as pd

# CSV (local ou URL)
df = pd.read_csv('fichier.csv')

# Excel
df = pd.read_excel('fichier.xlsx', sheet_name='Feuille1')

# Base SQL
import sqlite3
conn = sqlite3.connect('base.db')
df = pd.read_sql_query("SELECT * FROM table", conn)

# API JSON
import requests
df = pd.DataFrame(requests.get('https://api.exemple.com/data').json())
```

Pandas : Exploration et Manipulation

Explorer les données

```
# Aperçu des données
print(df.head())      # 5 premières lignes
print(df.info())      # Types et valeurs manquantes
print(df.describe())  # Statistiques descriptives

# Sélection et filtrage
df['colonne']          # Sélectionner une colonne
df[['col1', 'col2']]   # Plusieurs colonnes
df[df['age'] > 25]      # Filtrer les lignes
```

Manipuler les données

```
# Ajouter/modifier des colonnes
df['nouvelle_col'] = df['col1'] + df['col2']

# Supprimer des colonnes ou lignes
df.drop(columns=['col1'], inplace=True)
df.dropna() # Supprimer les lignes avec valeurs manquantes
```

Introduction aux Datasets

- Un **dataset** est un ensemble de données utilisé pour entraîner, valider et tester les modèles d'apprentissage automatique.
- Les datasets sont essentiels pour la modélisation et la généralisation des modèles de Machine Learning.
- **Exemples de datasets courants :**
 - Iris (classification des fleurs)
 - MNIST (chiffres manuscrits)
 - California Housing (prédiction des prix des logements)

Utilisation des Datasets avec sklearn

- La bibliothèque **scikit-learn (sklearn)** propose de nombreux datasets standards pour l'entraînement des modèles.
- Les datasets peuvent être chargés via `sklearn.datasets`.
- Les types de datasets incluent :
 - **Datasets intégrés** (Iris, Digits, etc.)
 - **Datasets de flux** (`fetch_*`) pour des jeux de données plus volumineux
 - <https://scikit-learn.org/1.5/api/sklearn.datasets.html>

Dataset Iris

- **Iris** : Dataset de classification des fleurs en trois espèces basées sur la longueur et la largeur des sépales et des pétales.

iris setosa



petal

sepal

iris versicolor



petal

sepal

iris virginica



petal

sepal

Figure: Source web

Chargement du Dataset Iris

```
from sklearn.datasets import load_iris
import pandas as pd

# Charger le dataset Iris
iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)

# Afficher les 5 premières lignes
print(df.head())
```

Chargement du Dataset California Housing

- **California Housing** : Données sur les prix des logements en Californie en fonction de plusieurs caractéristiques socio-économiques.

```
from sklearn.datasets import fetch_california_housing
import pandas as pd

# Charger le dataset California Housing
housing = fetch_california_housing()
df = pd.DataFrame(housing.data, columns=housing.feature_names)
df['Price'] = housing.target    # prix median, en 100k$

# Afficher les 5 premières lignes
print(df.head())
```

Dataset MNIST

- **MNIST** : Images de chiffres manuscrits (0-9) utilisées pour la reconnaissance de chiffres.

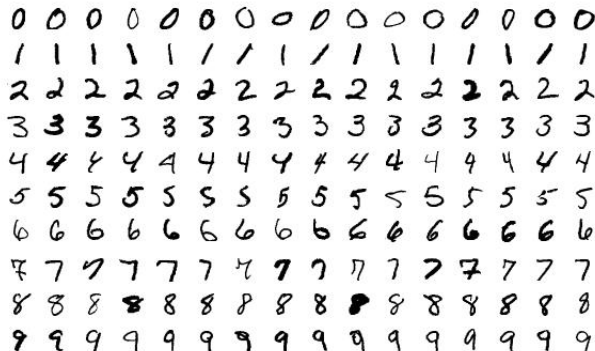


Figure: Source web

Structure des Datasets sklearn

- Les datasets de **sklearn** sont généralement des objets de type `Bunch`, similaires à des dictionnaires Python.
- Les principaux éléments d'un dataset **sklearn** incluent :
 - **data** : Tableau 2D ($n_{\text{échantillons}} \times n_{\text{caractéristiques}}$) contenant les données d'entrée.
 - **target** : Tableau 1D ou 2D contenant les étiquettes ou les cibles.
 - **feature_names** : Noms des caractéristiques ou des colonnes (optionnel).
 - **target_names** : Noms des classes ou des cibles (pour les tâches de classification).
 - **DESCR** : Description textuelle du dataset.
- Ces composants permettent de comprendre et de manipuler facilement les données.

Examiner les données

Examiner la structure des données

Objectifs :

- 1 Savoir ce que contient un dataset avant d'en faire quoi que ce soit.
- 2 Repérer les types de colonnes, les valeurs manquantes et les valeurs suspectes.

Dans toute cette partie, nous travaillons sur **California Housing** --- celui que vous manipulerez en TP dans une heure.

Les quatre réflexes

- 1 `df.head()`
À quoi ressemblent les données ?
- 2 `df.info()`
Combien de lignes, quels types, quelles valeurs manquantes ?
- 3 `df.describe()`
Les ordres de grandeur sont-ils plausibles ?
- 4 `df.isnull().sum()`
Où manque-t-il des données ?

Pourquoi c'est la première chose à faire

Un modèle entraîné sur des données qu'on n'a pas regardées produit des résultats qu'on ne sait pas interpréter.

La plupart des erreurs en apprentissage automatique ne viennent pas de l'algorithme, mais de données mal comprises.

1. head() --- à quoi ressemblent les données ?

```
import pandas as pd
from sklearn.datasets import fetch_california_housing
```

```
data = fetch_california_housing(as_frame=True)
df = data.frame
print(df.head())
```

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	MedHouseVal
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23	4.526
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22	3.585
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24	3.521
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25	3.413
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25	3.422

Une ligne n'est pas un logement, mais un **groupe de blocs** (environ 1 500 habitants).

MedHouseVal, la cible, est en **centaines de milliers de dollars**.

2. info() --- types et valeurs manquantes

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 9 columns):
#   Column          Non-Null Count  Dtype
---  -
0   MedInc          20640 non-null  float64
1   HouseAge        20640 non-null  float64
2   AveRooms        20640 non-null  float64
3   AveBedrms       20640 non-null  float64
4   Population      20640 non-null  float64
5   AveOccup        20640 non-null  float64
6   Latitude        20640 non-null  float64
7   Longitude       20640 non-null  float64
8   MedHouseVal     20640 non-null  float64
dtypes: float64(9)
memory usage: 1.4 MB
```

Ce qu'on lit ici :

- **20 640** observations, **9** colonnes
- Toutes numériques (float64) :
aucun encodage à prévoir
- **20 640 non-null** partout :
aucune valeur manquante

Attention

Un objet dans la colonne `Dtype` signale du texte --- souvent un nombre mal lu, à convertir.

3. describe () --- les ordres de grandeur sont-ils plausibles ?

	AveRooms	AveBedrms	AveOccup	MedHouseVal
count	20640.000000	20640.000000	20640.000000	20640.000000
mean	5.429000	1.096675	3.070655	2.068558
std	2.474173	0.473911	10.386050	1.153956
min	0.846154	0.333333	0.692308	0.149990
50%	5.229129	1.048780	2.818116	1.797000
max	141.909091	34.066667	1243.333333	5.000010

À vous : trois choses clochent dans ce tableau. Lesquelles ?

3. describe () --- ce que révèle une lecture attentive

	AveRooms	AveBedrms	AveOccup	MedHouseVal
mean	5.429000	1.096675	3.070655	2.068558
std	2.474173	0.473911	10.386050	1.153956
50%	5.229129	1.048780	2.818116	1.797000
max	141.909091	34.066667	1243.333333	5.000010

- ❶ **141 pièces** en moyenne par logement, et **34 chambres** : des valeurs aberrantes, ou des blocs très particuliers.
- ❷ **1 243 occupants** par logement, pour une médiane à 2,8 : l'écart-type (10,4) est plus de trois fois la moyenne.
- ❸ MedHouseVal s'arrête à **5,00001** : le prix est **plafonné** à 500 001 \$. Toutes les maisons plus chères ont été ramenées à cette valeur.

La leçon

Le point 3 est invisible pour un algorithme : votre modèle ne pourra **jamais** prédire au-delà de 500 000 \$, et vous ne le saurez que si vous avez regardé vos données.

4. Valeurs manquantes et valeurs aberrantes

Valeurs manquantes

```
print(df.isnull().sum())
```

```
MedInc          0  
HouseAge        0  
AveRooms         0  
...  
MedHouseVal     0
```

Ici : aucune. C'est confortable, et c'est rare — le dataset a déjà été nettoyé.

Valeurs aberrantes

```
import matplotlib.pyplot as plt
```

```
df['AveOccup'].plot(kind='box')  
plt.show()
```

Le **boxplot** rend visible d'un coup d'œil ce que `describe()` laissait deviner. Nous y reviendrons en détail la séance prochaine.

Quatre commandes, cinq minutes. **Toujours** avant de modéliser.

Ce qu'il faut retenir de cette séance

- ❶ L'apprentissage automatique consiste à **laisser les données produire le programme**, au lieu d'écrire toutes les règles à la main.
- ❷ Un modèle apprend en trois temps : **entraînement, validation, test**. L'objectif est de bien prédire sur des données **jamais vues**.
- ❸ Une donnée est **quantitative** (discrète ou continue) ou **qualitative** (nominale ou ordinale). Le traitement diffère selon le cas.
- ❹ La qualité des données détermine la qualité du modèle : **pertinence, fiabilité, représentativité, quantité**.
- ❺ La préparation des données suit **9 étapes**, qui structureront les séances 1 et 2.
- ❻ Avant toute chose : `head()`, `info()`, `describe()`, `isnull().sum()`.
On regarde ses données avant de modéliser.

Et maintenant : le TP

Au travail :

- 1 Installer l'environnement (Anaconda, environnement `ml`)
- 2 Lancer Jupyter Notebook
- 3 Charger *California Housing* depuis Scikit-learn
- 4 Examiner sa structure et répondre aux questions

Si vous terminez en avance

L'exercice bonus sur le dataset *Iris*, qui nous servira de fil rouge pour la classification.

La séance prochaine

Préparer les données :

- Visualiser (histogrammes, boxplots, nuages de points)
- Nettoyer (valeurs manquantes, doublons, aberrantes)
- Encoder les variables qualitatives
- Normaliser
- Séparer entraînement / validation / test

Notations utilisées dans le cours

Données	
$x_i, \mathbf{x}$	variable d'entrée, vecteur d'entrée
$y, \hat{y}$	valeur réelle, valeur prédite
n	nombre d'observations
p	nombre de variables explicatives
Paramètres du modèle	
β_0, β_j	ordonnée à l'origine, coefficients de régression (ch. 2)
$w_i, \mathbf{w}$	poids d'un neurone (ch. 3)
b	biais (ch. 3)
∂	paramètres d'un modèle

Réseaux de neurones	
z	pré-activation : $z = \sum_i w_i x_i + b$
f, σ	activation, sigmoïde
$a^{(l)}$	activations de la couche l
$W^{(l)}, b^{(l)}$	poids et biais de la couche l
Apprentissage	
$J, \mathcal{L}$	fonction de coût, de perte
η	taux d'apprentissage
∇J	gradient de J

Les coefficients β de la régression et les poids w d'un neurone désignent la même chose : les paramètres ajustés pendant l'apprentissage. Seule la tradition de notation diffère.